Série de TP n° 1

Exercice 1 : Ecrire un programme qui vérifie que les parenthèses d'une chaîne sont bien équilibrées.

Solution: **exo1.cpp**

```
#include<stdio.h>
#include<conio.h>  // pour getch()
#include<string.h>

int main()
{
   int i,           // indice pour parcourir la chaine
      j=0;        //entier qui s'incrémente s'il rencontre '(' et se decremente si ')'
   char c,       // le caractère courant
       ch[30]; //la chaine lue au clavier

    printf("Donnez votre chaine a analyser:\t");
    gets(ch);
    i=0;

    while (i<strlen(ch)) //boucle qui teste la fin de la chaine
    {
         c=ch[i];
         if (c=='(') ++j;
         else if (c==')') --j;
         i++;
    }
    if(!j) printf("les parenthses sont bien equilibrees");
    else if(j<0) printf("il manque %d  parenthese ouvrante",-j);
        else printf("il manque %d parenthese fermante",j);

    getch();
}
```

Exercice 2 : A l'aide d'une déclaration de type ***union***, écrire une fonction CTOA, qui lit un caractère et renvoie le code ascii correspondant dans son deuxième argument.

Solution : **exo2.cpp**

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream.h>
union
    {
      char c;
      unsigned i;
    } p;
void CTOA()
{
    p.i=0;
    cout<<"Donner un caractere: ";
    cin>>p.c;
    cout<<"\n\tSon code ASCII est: "<<p.i;
    cout<<"\n";

}
int main()
{
   CTOA();
   system("pause");
    return 0;
}
```

Exercice 3 : Ecrire un programme ayant pour effet de synthétiser des expressions arithmétiques composées du seul chiffre 4, des quatre opérateurs +, -, *, / et les parenthèses (,).

Par exemple : on peut écrire : « 2 = (4+4)/4 » et « 5=4+(4/4) ».
On teste ce programme pour les entiers de 1 à 20.

Solution : **exo3.cpp**

```
#include<stdio.h>
#include<stdlib.h>

// Synthèse d'expressions avec le chiffre 4 les parenthèses et les 4 opérateurs

// la fonction exp reçoit un entier et donne une expression résultat
void exp(unsigned i)
{
    unsigned n1=i/4; // n1 la partie entière
    unsigned n2=i%4; // le reste de la division par 4
    // d'une manière récursive on refait la même chose à n1 jusqu'à trouver une
    // partie entière égale à 0
    if(n1!=0)
          {
                printf("4");
                if (n1>1) {printf("*(",n1); exp(n1); printf(")");}
          }
                if (n2!=0 && n1!=0) printf("+(");
                if (n2==1) printf("4/4");
                if (n2==2) printf("(4+4)/4");
                if (n2==3) printf("4-(4/4)");
                if (n2!=0 && n1!=0) printf(")");
   if(n1==0) return;
}

int main()
{
   for(unsigned i=1; i<=20; printf("\n%i= ",i), exp(i), i++);
   system("pause");
   return 0;
}
```

Exercice 4 : Ecrire un programme déterminant le nombre de la lettre ‘e’ (minuscule) dans un texte fourni au clavier.

Solution : **exo4.cpp**

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define CAR ‘e’
#define LGMAX 132

int main()
{  char texte[LGMAX+1];
   char *adr ;
   int ncar ;
   printf("donnez un texte terminé par return\n") ;
   gets (texte) ;
   ncar = 0 ;
   adr = texte ;
   while (adr=strchr(adr,CAR))
        { ncar++;
          adr++;
        }
  printf (“votre texte comporte %d fois le caractère %c”, ncar, CAR);
  system("pause");
  return 0;
}
```

Exercice 5 : Ecrire un programme qui supprime toutes les lettres ‘e’ (minuscules) d’un texte fourni au clavier.

Solution : **exo5.cpp**

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define CAR ‘e’
#define LGMAX 132

int main()
{  char texte[LGMAX+1];
   char *adr ;
   int ncar ;
   printf("donnez un texte terminé par return\n") ;
   gets (texte) ;
   adr = texte ;
   while (adr=strchr(adr,CAR)) strcpy(adr, adr+1);
   printf (“voici le texte sans les caractères %c %s\n”, CAR, adr);
   puts (texte) ;
   system("pause");
   return 0;
}
```

Exercice 6 : Ecrire un programme qui lit au clavier un mot d'au plus 30 caractères et l'affiche à l'envers.

Solution : **exo6.cpp**

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define NBCAR 30
int main()
{
   char nom[NBCAR+1] ;
   int i ;
   printf("donnez un nom d'au plus %d caractères : ", NBCAR) ;
   gets(nom) ;
   for (i=strlen(nom) ; i>=0 ; i--)
          putchar(nom[i]);
  system("pause");
  return 0;
}
```

Exercice 7 : Ecrire un programme qui lit un verbe du premier groupe et en affiche la conjugaison au présent, sous la forme :
je chante
tu chantes
il chante
nous chantons
vous chantez
ils chantent

Le programme devra vérifier que le mot fourni se termine par « er ».

Solution : **exo7.cpp**

```
 #include <stdio.h>
#include <string.h>
#include<conio.h>
#define LMAX 26
int main()
{
        char verbe [LMAX+1];
        char *adfin;
        char *term[6] = {“e”, “es”, “e”, “ons”, “ez”, “ent”};
        char *deb[6] = {“je”, “tu”, “il”, “nous”, “vous”, “ils”};
        int i ;
        do
                {printf (“ donnez un verbe de premier groupe : “) ;
                 gets(verbe) ;
                 adfin = verbe + strlen(verbe) – 2 ; // voir la terminaison « er »
                }
        while (strcmp (adfin, “er“) ) ;
                printf(“ \n le verbe au présent  est:\n“) ;
                for (i=0 ; i<6 ; i++)
                        { strcpy(adfin, term[i]) ;
                          printf (“%s  %s\n “, deb[i], verbe) ;
                        }
getch();
return 0;
}
```